

Docket No. P9222

PATENT

UNITED STATES APPLICATION FOR LETTERS PATENT

for

**DYNAMIC REMOVAL OF A DRIVER STACK WHEN A PARENT DRIVER USES A
CHILD DRIVER**

By

Rajesh R. Shah

Filed

August 2, 2000

5 **DYNAMIC REMOVAL OF A DRIVER STACK WHEN A PARENT DRIVER USES A
CHILD DRIVER**

FIELD OF THE INVENTION

10 This invention pertains to computer operating systems, and more particularly to
dynamic removal of driver stacks from the operating system.

BACKGROUND OF THE INVENTION

15 Although computer systems begin with the hardware components, computer operation
requires software that controls the use of the hardware components. Typically handled by the
operating system running on the computer, these software elements are called *device drivers*
(sometimes shortened to *drivers*). The device drivers are typically specific to the hardware
component they support. Because accessing a particular hardware component requires
cooperation among multiple software drivers, those device drivers are grouped into *driver*
stacks (sometimes shortened to *stacks*). The drivers in each stack work together, allowing
20 access to the desired hardware component.

FIG. 1 shows software hierarchy 105 of software drivers to access a hard disk.
Software hierarchy 105 is typical of hierarchies for accessing a particular hardware
component. At the top of hierarchy 105 is the Peripheral Component Interconnect (PCI) bus
driver. Below that is the Small Computer System Interface (SCSI) adapter driver. The SCSI
25 adapter includes a SCSI port, onto which is connected the hard disk, which has a disk
partition on which data may be stored. For each of the devices in the hierarchy, device
drivers are needed to control access to the hardware devices. Below the SCSI adapter driver
is the SCSI port driver.

Each device driver may export services specific to that device driver. These services
30 may be used by other drivers in the driver stack or by other programs, such as the operating
system. FIG. 2A shows driver stack 205 with three device drivers 210, 215, and 220, each
device driver providing a service. Driver 215 is using the service provided by driver 210 (as
shown by arrow 225), and driver 220 is using the service provided by driver 215 (as shown
by arrow 230).

Each driver needs to know if other drivers are using its services. This information is important in case the driver stack is to be removed (see below). Because their services are being used, drivers 210 and 215 have non-zero reference counts. Note that it is not important for the driver to know who is using its service, only that its service is being used.

5 Although FIG. 2A shows only children drivers using the services of their immediate parents, this is not a limitation of driver services export. For example, in FIG. 2B, driver 210 in driver stack 205, is shown using the services exported by its "grandchild," driver 220 (shown by arrow 235).

10 It may happen that a driver stack, loaded to allow access to a particular hardware component, is no longer required. For example, the hardware component in question may have been removed from the computer system. When a driver stack can be removed from the operating system while the computer is in use, the driver stack is called *dynamic*. The operating system interrogates each driver in the stack, asking the drivers if they may be removed. Then, if each driver in the stack approves the remove query, the driver stack is removed. (Static driver stacks are also possible, but in a static driver stack, the driver stack
15 may not be removed while the computer is running. The static driver stack can only be "removed" by specifying it not be loaded when the computer is next started.)

FIGs. 3A and 3B show the procedure used to query whether a driver stack may be removed. At block 305, the lowest device in the driver in the stack receives the remove query. At decision point 310, the device checks to see if its services are being used by any other programs. If its services are being used, then at block 315 the device signals the operating system that the driver stack may not be removed. Otherwise, at decision point 320 the device checks to see if it is using any services provided by other devices. If it is, then at block 325 the device stops using the service, and at block 330 the device decrements the
25 reference count of the devices whose services it was using. At decision point 335, the device checks to see if it has a parent device in the driver stack. If it does, then at block 340 the device passes the remove query to its parent device, and the process returns to block 305. Otherwise, the last device in the driver stack has approved the remove query, and at block 345 the device signals the operating system that the driver stack may be removed.

30 A person skilled in the art will recognize that the procedure of FIGs. 3A and 3B will fail when a parent device uses the services of a child device, as shown in FIG. 2B, because the child device must approve the remove query before the parent device receives notice of the remove query. Referring to FIG. 2B, child driver 220 will note that its services are being

used and immediately fail the remove query. Parent driver 210 never has a chance to stop using the services of child driver 220, allowing the remove query to succeed.

The present invention addresses these and other problems associated with the prior art.

5

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 shows a hierarchy of device driver object components in a computer system requiring a driver stack.

FIG. 2A shows a typical driver stack.

10 FIG. 2B shows the driver stack of FIG. 2A with a parent driver accessing a service of a child driver.

FIGs. 3A and 3B show the procedure used by an operating system to remove a dynamic driver stack.

15 FIG. 4 shows a computer system running an operating system that may dynamically remove driver stacks in accordance with the invention.

FIG. 5 shows the driver stack of FIG. 2B using a virtual device object in accordance with the invention.

FIG. 6 shows a virtual device object in accordance with the invention redirecting a remove query to the parent driver in the operating system of FIG. 4.

20 FIG. 7 shows the procedure used to enable a parent device to access a service provided by a child device in the operating system of FIG. 4 in accordance with the invention.

FIG. 8 shows the procedure used by the virtual device to process a remove query received from the operating system of FIG. 4 in accordance with the invention.

25

DETAILED DESCRIPTION

FIG. 4 shows a computer system 405 in accordance with the invention. Computer system 405 includes a computer 410, a monitor 415, a keyboard 420, and a mouse 425.

30 Computer 410 includes hardware components, such as a central processing unit, a memory, and a cache (not shown). Computer system 405 may also include other equipment not shown in FIG. 4, for example, other input/output equipment or a printer.

Running on computer system 405 is operating system 430. Operating system 430 includes drivers, such as drivers 435-1, 435-2, 435-3, 435-4, and 435-5. These drivers are

organized into driver stacks, such as 440-1 and 440-2. Note that driver stacks 440-1 and 440-2 are for exemplary purposes only. There may be more than two driver stacks loaded in operating system 430, and driver stacks 440-1 and 440-2 may have differing numbers of drivers within each driver stack. A person skilled in the art will also recognize that drivers may be duplicated in multiple driver stacks: for example, driver 435-2 is included in both driver stacks 440-1 and 440-2.

FIG. 5 shows how driver stack 205 of FIG. 2B may be implemented according to the invention to allow for the removal of driver stack 205. In FIG. 5, parent driver 210 creates virtual device object 505. Virtual device object 505 is inserted into driver stack 205 below the accessed child device 220. In practice, virtual device object 505 will be placed at the bottom of driver stack 205. But if the implementation allows, virtual device object 505 may be inserted anywhere in driver stack 205, provided virtual device object 505 is below the accessed child device 220. Virtual device object 505 then accesses the service of child device 220 on behalf of parent device 210, as shown by dashed line 510. Virtual device object 505 is "bound" to parent device 210, as shown by line 515. In effect, virtual device object 505 is a "placeholder" for parent device 210. Parent device 210 "fools" child device 220 into thinking child device 220 is being accessed by a lower child device.

By creating virtual device object 505 and placing it in the driver stack below child device 220 whose services are being accessed, the standard remove query procedure of FIGs. 3A and 3B may be used. The remove query will be delivered to virtual device object 505 before it is delivered to child device 220. Virtual device object 505 then informs parent device 210 of the remove query. Parent device 210 can stop using the service of child device 220. Virtual device 505 may then approve the remove query and pass the remove query to child device 220. Because parent device 210 is no longer using the services of child device 220, child device 220 may also approve the remove query. The remove query iterates up the driver stack, and ultimately may be approved by every driver in the driver stack. The operating system is then able to remove the driver stack.

A person skilled in the art will recognize that this technique may be extended beyond the case of a single parent device accessing services of a single child device. Each parent device that wants to access a service of a child device may add a virtual device object to the driver stack below the accessed child device. Further, a single parent device may access the services of multiple child devices using a single virtual device object, provided the virtual

device object is below all the child devices whose services are being used by the parent device.

It may happen that a new child device is added to the driver stack below the virtual device, the new child device providing a service desired by the parent device. The parent device may either add a new virtual device object or relocate the existing virtual device
5 below the new child device in the driver stack. (In practice, it is preferable for the parent device to add a new virtual device object below the new child device and use the new virtual device object only for accessing the services of the new child device.)

Because the virtual device object is a placeholder for the parent device, the parent
10 device is actually accessing the service of the child device. When the virtual device object receives a remove query from the operating system, the virtual device object lets the parent device know that the parent device should stop using the services of the child device. Referring to FIG. 6, since virtual device object X 505 is "bound" to driver A 210, when the operating system 430 sends remove query 605 to device object X 505 it is actually sending
15 remove query 605 to driver A 210. Driver A 210 can then stop using the services of the child device, and virtual device 505 may pass remove query 605 to the next device driver in the driver stack.

There are several ways that virtual device 505 can inform parent device 210 about remove signal 605. One way is to use events, as described in object-oriented programming.
20 Another way is for the operating system to directly invoke the code in the virtual device object. This code may directly link to code in the parent device for processing remove queries. A person skilled in the art will also recognize other techniques for passing the remove query from the virtual device object to the parent device.

FIG. 7 shows the procedure used to enable a parent device to access a service
25 provided by a child device in the operating system of FIG. 4 in accordance with the invention. At block 705, the new virtual device object is created. At block 710, the virtual device object is bound to the parent device. At block 715, the virtual device object is inserted into the driver stack below the child device whose services are sought. At block 720, the parent increments the reference count of the child device. Finally, at block 725, the parent
30 device accesses the services of the child device.

FIG. 8 shows the procedure used by the virtual device to process a remove query received from the operating system of FIG. 4 in accordance with the invention. At block 805, the virtual device receives the remove query. At block 810, the parent device stops using the

services of the child device. At block 815, the parent device decrements the reference count of the child device. Finally, at block 820, the virtual device object passes the remove query to the next device in the driver stack.

5 Having illustrated and described the principles of my invention in an embodiment thereof, it should be readily apparent to those skilled in the art that the invention can be modified in arrangement and detail without departing from such principles. I claim all modifications coming within the spirit and scope of the accompanying claims.

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000